

Chapter 12

Floreon⁺ Modules: A Real-World HARPA Application in the High-End HPC System Domain



Antoni Portero, Radim Vavrik, Martin Golasowski, Jiri Sevcik, Giuseppe Massari, Simone Libutti, William Fornaciari, Stepan Kuchar, and Vit Vondrak

12.1 Floreon⁺: HPC Domain Application

There are many types of natural disasters in the world, and many of them depend on the geographical location of the observed area. Floods are one of the worst and most recurrent types of natural disasters in Central and Eastern Europe. Floods [17] occur when discharges and water levels exceed their bank-full discharge values and overflow. Floods kill millions of people, more than any other natural disaster, and they are also the world's most expensive type of natural disaster.

Floods frequently affect the population and are therefore studied by numerous scientific research institutes. Almost all large rivers in Central and Eastern Europe have experienced catastrophic flood events, e.g., the 1993 and 1995 flooding of the river Rhine, the rivers Danube and Theiss in 1999 and 2002, the river Odra in 1997, the river Visla in 2001, and the river Labe in 2002. Floods, however, affect not only Central and Eastern Europe, but they represent a significant problem in many regions all around the world. The growing number of losses caused by floods in countries around the world suggests that global mitigation of disasters is not a simple matter, but rather a complex issue in which science and technology can play a significant role. Therefore, the issue of flood prediction and simulation has been selected as a case of choice for innovative development.

A. Portero (✉) · R. Vavrik · M. Golasowski · J. Sevcik · S. Kuchar · V. Vondrak
IT4Innovations, Ostrava - Poruba, The Czech Republic
e-mail: antonio.portero@vsb.cz; radim.vavrik@vsb.cz; martin.golasowski@vsb.cz;
jiri.sevcik@vsb.cz; stepan.kuchar@vsb.cz; vit.vondrak@vsb.cz

G. Massari · S. Libutti · W. Fornaciari
Politecnico di Milano, Milano, Italy
e-mail: giuseppe.massari@polimi.it; simone.libutti@polimi.it; william.fornaciari@polimi.it

The project FLOREON⁺ [15] started in 2006 as a research project funded by the regional government of the Moravian-Silesian region of the Czech Republic that required reliable models for flood simulations and predictions to minimise costs of post-flood repairs in areas impacted by severe floods. The principal goal of the research project FLOREON⁺ (*FLOods REcognition On the Net*) is the development of a prototype modular system of environmental risks modelling and simulations. FLOREON⁺ is based on modern Internet technologies with platform independence.

The FLOREON⁺ project results help to simplify the process of disaster management and increase its operability and effectiveness. The main scopes of modelling and simulation are flood risk, traffic modelling during critical situations, water and air pollution risks, and other environmental hazards. Another efficient utilisation of the computing power could be computing What-If scenarios for decision support and to plan preventive actions. Part of the research employs the modelling of land cover, and land use changes based on thematic data collection (aerial photographs, satellite imagery), and application of the prediction tools brings attractive advantages to land use planning. Modelling the catchment response to severe flood events generates the opportunity to improve the set-up and dimensions of new channel systems, within the scope of hydrology and water management.

12.2 Experimental Application

The central thematic area of the project is hydrologic modelling and prediction. The system focuses on acquisition and analysis of relevant data in near-real time and uses this data to run hydrologic simulations with a short-term forecast. The results are then used for decision support in disaster management processes by providing predicted discharges on river gauges and prediction and visualisation of inundated areas in the landscape (Fig. 12.1).

In the simulation phase (Adaptivity in Simulations [2]) of the prediction cycle, adaptivity in the spatial resolution is essential to improve the accuracy of the result. Specifically, more computational resources are introduced when the weather looks more attractive (i.e. after hours or days of heavy rain). These may be used for computations that are triggered by stimulating activities detected in the forecast simulation. Or they may be part of the same simulation process execution if it has been re-engineered to use automatic adaptive accuracy refinement. In any case, the most accurate computation has to track the evolution of the predicted and actual weather in real time. The location and extent of finer results should evolve and move across the simulated landscape in the same way the actual weather is constantly moving.

The model used in this study is a rainfall–run-off model (RR) developed as part of the Floreon⁺ project. Rainfall–run-off models [14] are dynamic mathematical models which transform rainfall to flow at the catchment outlet. The primary purpose of the model is to describe rainfall–run-off relations of a catchment area. Standard inputs of the model are precipitation measured by meteorological gauges,



Fig. 12.1 Inundation area for the Ostravice river

and spatial and physical properties of the river catchment area. Common outputs are surface run-off hydrographs which depict relations between discharge (Q) and time (t). Catchment areas for the model used for experiments in this chapter are parameterised by Manning's roughness coefficient (N), which approximates physical properties of the river basin, and CN curve values (CN), which approximate geological properties of the river catchment area.

The confidence intervals are constructed using the Monte Carlo (M-C) method where input data sets are sampled from probability distributions extracted from historical results and used as input for a large number of simulations. The modelling precision of the model uncertainty can be positively affected by increasing the number of M-C samples and can be determined by estimating the Nash–Sutcliffe model efficiency coefficient [18] between the original simulation output and one of the percentiles selected from the Monte Carlo results.

Only selected percentiles are extracted from all M-C simulation results and saved on the platform. These percentile simulations describe a possible development of the situation taking possible inaccuracies into account along with its probability. For example, 80th percentile specifies that there is an 80th probability that the real river discharge will be lower or equal to the simulated discharge based on historical data. These results can then be propagated further into the flood prediction process, for example, used as input for hydrodynamic modelling. Figure 12.2 shows the visualisation of simulated inundated areas based on RR uncertainty results.

The confidence intervals' accuracy can be positively affected by increasing the number of MC simulations (also referred to as *samples*), and can be determined by estimating the Nash–Sutcliffe model efficiency coefficient [18] between the original simulation output and one of the percentiles selected from the Monte Carlo results. The percentile simulations describe a possible development of the situation taking possible inaccuracies (along with their probability) into account.

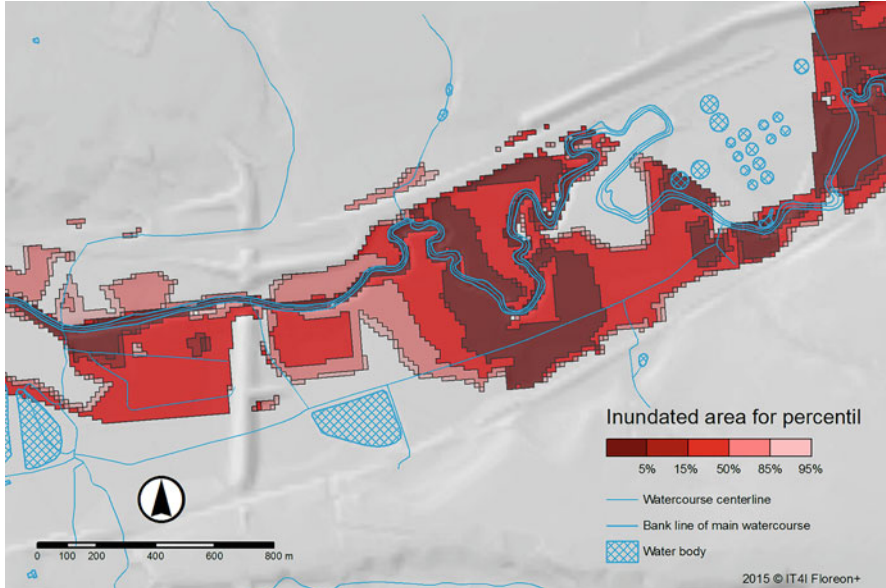


Fig. 12.2 Inundated area for percentile

12.2.1 Flood Warning Process

The flood warning process [24] can have two distinct stages: flood warning and response. The flood warning stage starts with a detection of the potential flooding threat. This activity should be done periodically based on river and precipitation monitoring and meteorology forecast. When a threat is detected, the flood warning committee is informed, and it issues a request for more detailed forecasts to the Institute of Hydrometeorology and local catchment area offices. These organisations provide

- Information about the actual river and reservoir situation
- Rainfall–run-off (RR) modelling simulation of surface run-off
- Hydrodynamic (HD) modelling flood lake simulations, flood maps, simulations of water elevation and water velocity, a real-time hydrological model for flood prediction using GIS, sediment transport, water quality analysis, etc.
- Erosion modelling simulation of water erosion
- Collection and archiving of flood data that can be used for estimating the magnitude of the flood based on historical evidence

If these predictions identify possible emergency situations, the flood warning committee alerts relevant agencies, and the process moves to the response stage. In this stage, countermeasures are implemented based on the forecast simulations from the flood warning stage. Also, flood predictions are still provided even

in the response stage to support the decision processes related to performed countermeasures and actions. During the response stage, additional areas can be affected by the flood emergency, and these predictions should be able to identify these areas in advance.

We have integrated the most computationally demanding module of the Floreon⁺ platform with HARPA-OS to examine how HARPA-OS will influence its execution. The selected module provides uncertainty modelling of rainfall–run-off (RR) models. One of the inputs of such models is the precipitation forecast computed by numerical weather prediction models that can be affected by particular inaccuracies.

12.2.2 The Flood Forecasting Model

The RR models transform precipitation to water discharge levels by modelling individual parts of the rainfall–run-off process. Common inputs of these models are an approximation of physical properties of modelled river catchment and a time series of precipitations. Outputs of these models are represented by a time series of water discharge levels (i.e. the relation between water discharge (Q) and time (t)) for modelled parts of the river course. Our RR model uses the SCS-CN method [5] for transforming rainfall to run off with the main parameter curve value (CN) approximated from the hydrological soil group, land use, and hydrological conditions of the modelled catchments. The contribution from river segments to a sub-basin outlet is computed using the kinematic wave approximation parameterised by Manning’s roughness coefficient (N), which approximates physical properties of the river channel using Manning’s roughness coefficient (N).

$$E = 1 - \frac{\sum_{t=1}^T (Q_m^t - Q_s^t)^2}{\sum_{t=1}^T (Q_m^t - \overline{Q_s^t})^2} \quad (12.1)$$

Where:

Q_m is the measured discharge in a specific time-step.

Q_s is the simulated discharge in a specific time-step.

E is the Nash–Sutcliffe model efficiency coefficient.

$E = 1$ means that the simulation matches observed data perfectly.

$E = 0$ means that the simulation matches median of the observed data.

$E < 0$ means that the simulation is less precise than the median of the observed data.

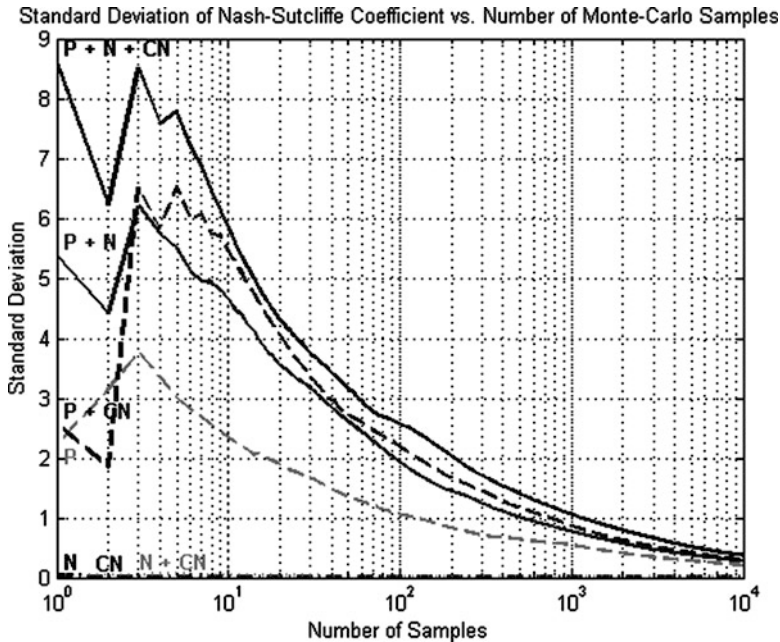


Fig. 12.3 Comparison of the precision provided by standard deviation of Nash–Sutcliffe coefficient and the number of Monte Carlo samples for different model parameters and their combinations

12.2.2.1 Uncertainties of the Rainfall–Run-off Model

The input precipitation data for short-term prediction are provided by numerical weather prediction models and can be affected by certain inaccuracies. Such inaccuracies can be projected into the output of the model by constructing confidence intervals. These intervals provide additional information about possible uncertainty of the model output and are constructed using the Monte Carlo (M-C) method. Data sets are sampled from the model input space and used as input for a large number of simulations. Precision of uncertainty simulations can be affected by changing the number of provided M-C samples. The precision of the simulations can be determined by computing the Nash–Sutcliffe (*NS*) model efficiency coefficient between the original simulation output and one of the percentiles selected from the Monte Carlo results. The *NS* coefficient is often used for estimating the precision of a given model by comparing its output with the observed data, but it can also be used for comparison between any two model outputs.

Figure 12.3 shows the precision of the simulated uncertainty results based on the number of Monte Carlo samples and different combinations of modelled parameters in the experimental model. *P* shows the standard deviation of uncertainty results where the only uncertainty of input precipitations was taken into account, *N* stands for the uncertainty model of the Manning’s coefficient, *CN* for the *CN*

uncertainty model, $N + CN$ describes the combination of Manning's coefficient and CN uncertainty models, and so on. These results show that the standard deviation is unstable for low numbers of M-C samples but starts to decrease steadily from around 5–8 samples. Precipitation uncertainty models and their combinations also show much higher standard deviations than CN and N based models. To obtain a sufficient precision for the simulation (i.e. minimise standard deviation of the result), the number of Monte Carlo samples for uncertainty simulation of the used Rainfall–Run-off model has to be in the order of 10,000 to 100,000. The sample count depends on the number of simulated model parameters and complexity of the model. Uncertainty simulation of CN and N could be executed for a much smaller sample count while maintaining sufficient levels of precision. This was mainly due to the lower sensitivity of the CN and N parameters when compared to the precipitation parameter, and also to the fact that precipitation uncertainty was sampled for each time-step of the simulation and each observed gauge, while CN and N did not depend on time.

12.2.2.2 On-Demand Simulations

Under the on-demand hydrologic simulations, a framework for running on-demand What-If Analysis (WIA) is created to simulate crisis situations. This also includes What-If hydrologic simulations. Through the web interface, users can create their own hydrologic What-If simulation running on this framework. They must choose the basic settings from the menu having the option to select river basin, schematisation, and rainfall–run-off model, for which the rainfall–run-off simulation and hydrodynamic model will be calculated. The framework allows the user to specify precipitation in selected precipitation stations and at selected times. This type of simulation also allows the user to edit the default parameters of sub-basins and channels. The next step is to run the simulation execution. Execution of What-If simulations is processed on an HPC cluster. Because the rainfall–run-off model is used as an input for the hydrodynamic model, the system allows the users to view hydrographs as soon as they are available, independently of the hydrodynamic computation. As soon as the hydrodynamic model computation is completed and the result values are stored within the spatial-temporal database, the users can view the simulated flood layer within the map interface together with the hydrographs.

12.2.3 Catchments Simulation

The proposed scenario monitors at runtime the behaviour of four concurrent instances of the uncertainty module. Each of these instances models the RR uncertainty for a different catchment of the Moravian-Silesian region: the Opava, Odra, Ostravice and Olza catchments (see Fig. 12.4). The watersheds are ordered

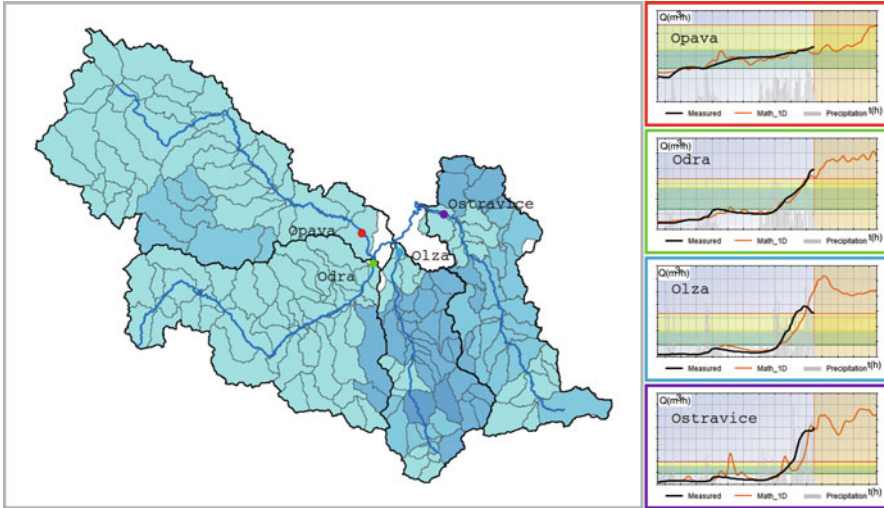


Fig. 12.4 The four main catchments (left) and outlet hydrographs (right) (the black line shows the measured discharge, the orange line shows the simulated discharge, X -axis: time in hours t (h), Y -axis: discharge, cubic meters per hour, Q (m³/h))

according to the impact in case of flooding (the lower the index, the higher the importance):

- C_1 : *Ostravice*—Functional urban areas with high population density and industrial areas in floodplain zones.
- C_2 : *Olza*—Flood-sensitive zones in urban areas.
- C_3 : *Odra*—Mountains in the upper part of the catchment can cause significant run-off. Less exposed urban areas.
- C_4 : *Opava*—Soils with low infiltration capacity.

Each catchment is simulated independently, and individual instances do not interact with each other.

12.2.4 Application Scenarios

Based on the weather, FLOREON⁺ can be subject to different service-level requirements. Indeed, the requirements can be translated to the parameters of the uncertainty modelling: A shorter response time in critical situations can, for example, be acquired by decreasing the number of MC samples—which, however, means reducing the precision of the results—or by allocating more computational resources to the application. Depending on the flood emergency situation, we

identified three application scenarios that have different requirements. According to their criticality level, we tagged the scenarios as *standard*, *intermediate*, and *critical*.

12.2.4.1 Standard Operation

In this scenario, the weather is favourable, and the flood warning level is below the critical threshold. In this case, the computation can be relaxed, and some errors and deviations can be allowed. The system should only use as much power as needed for standard operation. Only one batch of Rainfall–Run-off simulations with uncertainty modelling has to be finished before the next batch starts. The results do not have to be available as soon as possible, so no excessive use of resources is needed. In this case, the estimated accuracy can be reduced.

12.2.4.2 Intermediate Operation

Due to the presence of limited precipitations, the forecast of discharge exceeds a warning threshold. In this case, in order to decrease the uncertainty of the model, the number of MC samples that must be performed by the simulation increases.

12.2.4.3 Critical Operation

Several days of continuous rain raise the water in rivers or a very heavy rainfall on a small area creates new free-flowing streams. These conditions are signalled by the river water level exceeding the flood emergency thresholds or precipitation amount exceeding the flash flood emergency thresholds. Much more accurate and frequent computations are needed in this scenario, and results should be provided as soon as possible. The number of MC samples increases, to decrease the uncertainty of the model.

12.3 HARPA-OS for HPC Environments

The HARPA-OS is a runtime resource manager (as presented in Chap.4). Its role is to manage system resource allocation, taking into account both the status of the system resources and application requirements, combining pro-active and reactive strategies. For applications, performance requirements can vary not only among different applications but also during the execution of the same application. Aforementioned is a common scenario for HPC systems, where scientific applications make up most of the workload. It is a matter of fact that the performance requirements of this class of applications are often bound to the input data. This is due to the volume and type of data. For instance, monitoring systems acting

at preventing natural disasters need to execute applications implementing mathematical models steadily, to analyse input data, detect possible criticality and notify the needs of operating accordingly. To guarantee to the applications the required-level performance, a conventional approach to HPC systems is to statically reserve computational resources, isolating the execution environment through virtualisation techniques. However, to reserve a fixed amount of resources (e.g. processors or entire nodes) can be ineffective, with the applications owning resources that are not fully exploited, all the time. Scaling the problem to the dimensions of an HPC centre, with several applications to host, we can face an overall under-utilisation of the system resources, leading to two issues: (1) the fragmentation of the available resources, and thus less space for further applications; (2) the loss of efficacy of power management techniques. The latter is due to the fact without a proper *consolidation* of the computational resources to allocate, we can have processors, or single cores, which are not fully exploited, but use enough time to avoid them to go into a deep-sleep state, wasting power saving opportunities.

12.3.1 *The Runtime Resource Manager*

In the context of this work, we employed the HARPA-OS runtime resource manager [12, 16] (Chap. 4). HARPA-OS enables the management of multiple applications that compete for the usage of multiple many-core computation devices [8]. It also proposes a runtime library [3, 21] that is in charge of (a) synchronising the execution of applications with runtime-variable resource allocations, and (b) notifying the resource manager of the runtime-variable Quality of Service goals (QoS) of applications. So that the HARPA-OS scheduling policy, which can be either chosen from a set of predefined ones or implemented from scratch, can take into account the feedback coming from applications when computing resource allocations.

We designed and implemented PerDeTemp (PERformance DEgradation TEMPerature) Chap. 4, Sect. 4.6.2, a HARPA-OS scheduling policy that tries to meet the application performance requirements while minimising resource allocation [20]. When multiple computing resources are available, PerDeTemp employs a multi-objective heuristic to assign to applications only the healthiest and coolest cores. Such allocation aims at levelling the power flux over the whole chip, thus mitigating the ageing process and avoiding thermal hotspots [10].

Our framework is based on the idea of making the application terminate its execution just before the deadline (just-in-time, *jit*). This way, the amount of allocated processing elements is minimised. This, in turn, allows the resource manager to evenly level power consumption throughout the chip and to migrate the application to the coolest cores dynamically, thus evenly spreading heat and increasing the reliability of the silicon.

Figure 12.5 shows a comparison between the best-effort (maximises throughput and minimises execution time) of an application and our relaxed, deadline-aware

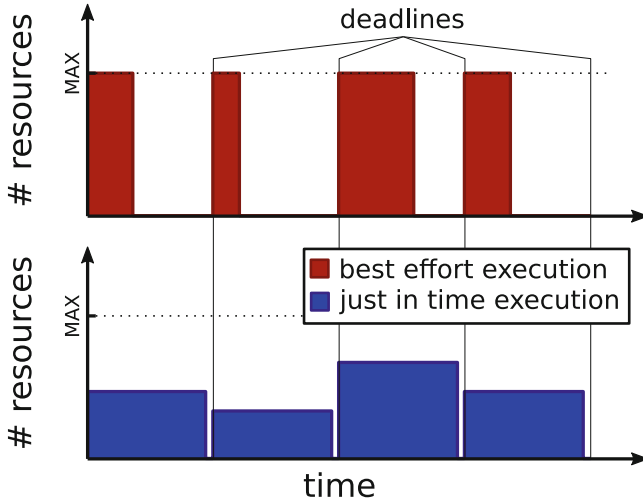


Fig. 12.5 Best-effort vs. *jit* scheduling

execution. Standard application execution is based on the idea of running the code as fast as possible (best-effort, *be*) using one thread per available processing element. This way, the results are available sooner; however, power consumption is maximised, and all the processing elements are stressed. Conversely, our just-in-time execution approach works with the idea that, since applications can send feedback to the resource manager dynamically, resource allocation can be made more elastic: It is adjusted over time so that the runtime-variable performance demand of applications is always complied with, but the execution time of applications is still the maximum allowed one (i.e. applications terminate just before their deadline). The resource manager exploits the now-unused resources as a resource pool that can be used in multiple ways, e.g., to provide cool cores when the next resource allocation is computed.

The HARPA-OS runtime, which is linked, manages applications, and transparently monitors application execution statistics. Amongst these, one of the most important ones is the average throughput. It is worth noticing that each time HARPA-OS changes the resource allocation of an application, the runtime re-sets the throughput statistics; hence, the average throughput computed by the runtime always refers to the current resource allocation. It follows, then, that the average throughput is a very accurate predictor of how the application will behave (i.e. whether the application will terminate or not before the deadline) if the resource allocation remains constant until the application termination.

The current resource allocation provides the scheduling policy with feedback; applications use the HARPA-OS runtime library API to retrieve their current execution time and their average throughput. Basing on those values, the applications can compare their current performance, i.e. the average throughput under the current

resource allocation, and the ideal throughput, i.e. the throughput that is needed by the application to terminate just before the deadline. This information is periodically sent back to the HARPA-OS as a performance gap:

$$\text{gap}_{\text{performance}} = \frac{\text{throughput}_{\text{current}} - \text{throughput}_{\text{ideal}}}{\text{throughput}_{\text{ideal}}} \quad (12.2)$$

While **positive performance gaps** mean that the application is executing too fast, which in turn indicates that the HARPA-OS may seize some of the allocated processing elements and insert them into the pool of empty resources, in contrast, **negative performance gaps** indicate that the application needs more resources. In this case, the HARPA-OS takes some of the less stressed (i.e. healthier—cooler) processing elements (i.e. cores) from the unused resources pool and adds them to the set of resources that can be exploited by the application. Finally, *performance gaps* equal to 0 mean that the application is likely going to terminate just in time. Even in this case, however, the HARPA-OS may decide to change resource allocation, usually to swap the currently allocated set of processing elements with healthier and cooler ones.

12.3.2 HARPA Integration

To fully exploit HARPA-OS, we implemented the application in a way that allows its reconfiguration during runtime. From the HARPA-OS side, the features of the application are a set of resource requirements. Fig. 4.3, from Chap. 4, Sect. 4.3, where it is described summarizes the manageable execution model and shows the different methods that have to be supported by the application:

- *onSetup*: Setting up the application (initialize variables and structures, starting threads, performance goal, etc.).
- *onConfigure*: Configuration/re-configuration code (adjusting parameters, parallelism, resources, number of active threads, etc.).
- *onRun*: Single cycle of computation (e.g. computing a single rainfall–run-off simulation for one Monte Carlo sample).
- *onMonitor*: Performance and QoS monitoring. Check the current performance with respect to the goal.
- *onRelease*: Code termination and clean-up.

12.3.3 Hardware Infrastructure

After integrating the HARPA methods into our flood forecasting models, we deployed the enabled applications to a part of a supercomputer. The supercom-

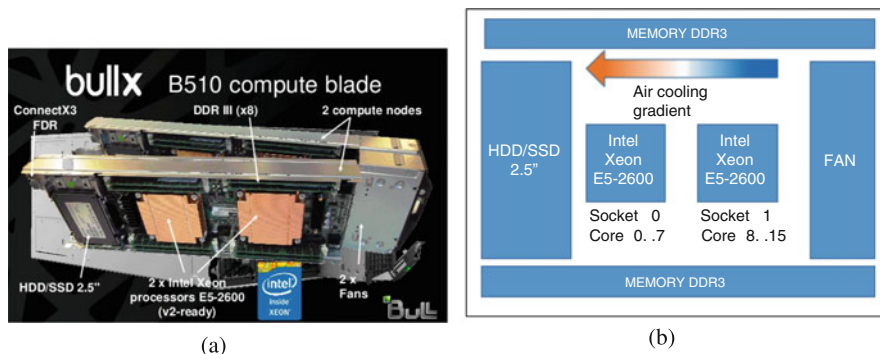


Fig. 12.6 The fan is on the right-hand side of the blade, creating an air cooling gradient between socket 0 and socket 1. (a) A photo of two Anselm Bullx B510 compute blades. (b) Schema of the blade

puter named Anselm (<https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview>, June 2013) is an HPC cluster operated by IT4Innovations, the Czech national supercomputing centre, which contains 180 computational nodes without accelerators (i.e. GPU, Xeon Phi, FPGA). All these nodes have an interlink of high-speed fully non-blocking fat tree InfiniBand and Ethernet networks.

In our experiments, we used a chassis of 18 blades; the nodes are connected through InfiniBand. Each Anselm node is an Intel Corporation Xeon E5/Core i7. Each node of the system consists of 16 cores—2 Intel(R) Sandy Bridge E5-2665 @ 2.4 GHz CPU sockets each with eight cores and 20480 KB L2 cache. All nodes equip at least 64 GB DRAM.

The blades in the cluster are the B510 model; Fig. 12.6 shows a simplified representation of their air cooling system. Given that the fan is not equally distant from the two sockets, one socket is cooler than the other one. When the system is idle, the temperature difference between the two sockets is approximately 10 °C. The system has several monitoring tools installed such as power meters, *ganglia* [25] and *likwid* [26].

Figure 12.6 shows the temperature of one blade when it is idle; there is nothing more than the operating system running on it. This fact is important because, during the temperature experiments, we observed differences of 10° between the best and the worst cooled areas, the worst being furthest from the fan. For these experiments presented in Sect. 12.3.4, we defined three working temperature zones. When the temperature is low, it is represented on the heat map in blue. When the temperature is average, at the proper level, it is labelled with green. And when the temperature starts to be high it is represented with yellow, and then when quite high with red. All the temperatures are always under the limit that the vendor (Intel) recommends. Figure 12.7 presents the heat map of an Intel socket with 16 cores. The X-axis represents time and the Y-axis the cores. The temperature is in Celsius degrees. The *Diff. Per time step* is the range of temperature between the coldest and hottest core in a specific time. The *Median* is the average temperature of the cores of the socket

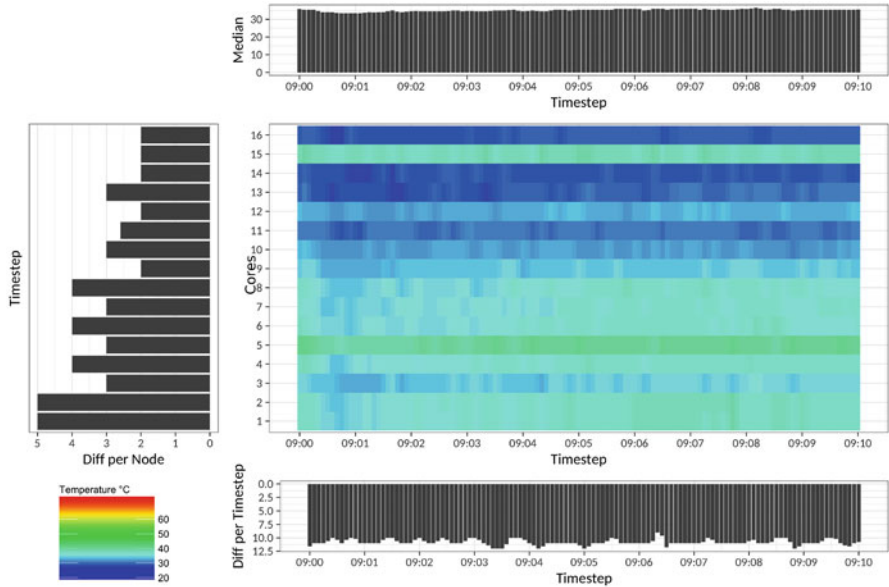


Fig. 12.7 Heat map of the two sockets of the blade in idle mode

in a specific time. The *Diff. per Node* is the maximum range of temperatures of a core during all the simulation.

12.3.4 Performance Analysis

The following experiments show the HPC solution to enforce runtime resource management decisions based on the standard control groups framework. A burst and a mixed workload analysis, performed on a multicore-based NUMA platform, reports some promising results both regarding performance, power saving, and ageing.

12.4 HARPA Testing in One Node

The following paragraph explains the execution of the models with HARPA-OS vs. non-HARPA-OS resource allocation. We run the models in the system with different governors. Governors that have a GNU/Linux Operating System (i.e. *performance* and *powersave*) by default set the frequency statically. The CPUfreq governor *powersafe* sets the lowest frequency from the `scaling_min_freq` border, and conversely the CPUfreq governor “*performance*” sets the highest frequency

from the `scaling_max_freq` border. These two governors give the power/energy consumption range from a maximum of the system to minimum power consumption. Other GNU/Linux governors exist such as *userspace* and *ondemand* that provide intermediate solutions. These governors are developed to give the best-effort solution (minimum execution time) for the given range concerning power and frequency. But what would happen if our constraint is the execution time?

12.4.1 Execution of the Models with Constraints

To answer the above question, we performed two experiments. The first one involved execution of uncertainties of the rainfall-run-off model in the HPC system. The implementation simulated a real case of 12K MC samples in 10 min. The QoS with HARPA-OS must minimise the thermal footprint of the scheme. Reduction of the power consumption and execution of the model had to be under the time constraints. The non-QoS execution had no information about power, temperature or execution time. The implementation of the design was achieved in three ways, two non-QoS (a and b), and another with QoS constraints (c):

- (a) Uncertainties of RR model in a non-managed HARPA-OS mode, only governed by the “performance” GNU/Linux governor.
- (b) Similar to (a) but with the “powersave” GNU/Linux governor.
- (c) Finally, uncertainties of RR model running in HARPA-OS [4] managed mode with the on-demand governor.

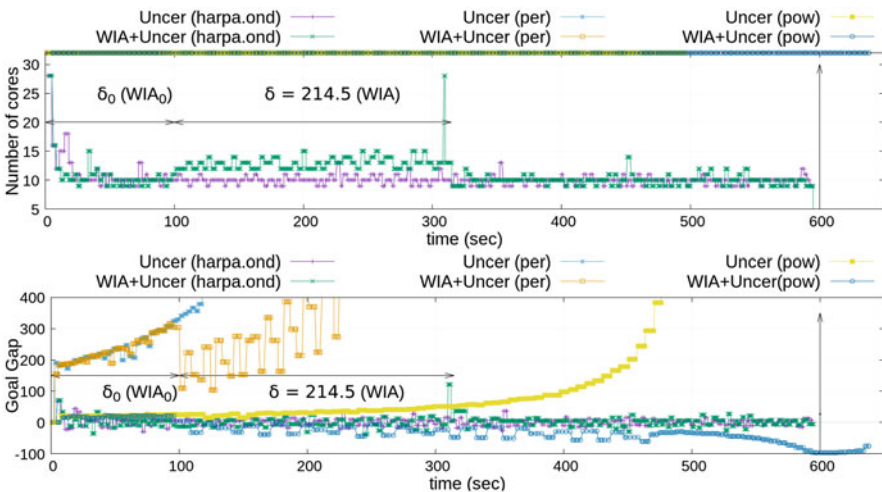


Fig. 12.8 Above: Number of resources allocated for performance, powersave and HARPA runtime with GNU/Linux ondemand governors during the execution of uncertainty models (*Uncer*) and Uncertainty and WIA concurrently (*Uncer+WIA*). Below: Goal gap during execution time for Uncertainty model vs. Uncertainty more WIA using the three governors (per, pow, HARPA)

The second part of the experiment is similar to the previous one. However, the second application, the WIA, starts running a delta time equal to 100 s and evaluates 20 possible (What-If) positions. During this period, δ the WIA is requesting all cores of the system. So, it is producing resource allocation constraints to the execution of the uncertainty model. Figure 12.8 presents the number of cores allocated by the different governors. In the case of GNU/Linux governors, this value is always 32, the maximum number of virtual cores that the system has. In the case of Harpa governor, this value changes dynamically trying the goal gap to be zero. Figure 12.8 shows the δ time when WIA is also executed in the system. During this period, the resources (cores allocated) to perform the uncertainty model in the system have increased. Figure 12.8 shows the goal gap. There are three cases:

- (a) *Performance*: The execution of the uncertainty model *Uncer(per)* ends long before the 10 min limit. Hence, the goal gap curve increases very quickly, even when the uncertainty and WIA run concurrently *WIA+Uncer(per)*. Then, during the δ time execution, there are some oscillations, which, however, go rapidly up since the model uses more resources than strictly necessary.
- (b) *Powersave*: In this case, there are two situations which are different to when there is the execution of uncertainty model alone, and when the uncertainty and WIA operate concurrently. In the first instance, it is possible to observe that the goal gap is quite near-zero for the first period of the simulation (first 300 s). Later, step by step, the goal gap starts to increase, which means that again the resources are more than enough to execute the model in a shorter time than 10 min. However, in the second case *WIA+Uncer(pow)*, the goal gap became negative in the last fifty seconds of the simulation, which means that the resources given to this governor are not enough to execute both models in 10 min as requested. So, the execution missed the time deadline.
- (c) *Harpa ondemand*: In this case, the runtime is minimising the goal gap as presented in section IV. The resources allocated (core #s are from 8 to 15, shown in Fig. 12.8) by the runtime are minimised as opposed to case (a) above, and the goal gap is oscillating around zero. In both cases, uncertainty (*Uncer(harpa.ond)*) and uncertainty more WIA (*Uncer+WIA(harpa.ond)*) are concurrently operating, and present similar behavior. The models finalise in the specified 10 min.

The curves presented in Fig. 12.8 show the different trade-offs of the monitors. In our case, the controls studied are power, energy consumption and temperature of the CPU sockets. Figure 12.9 presents the results related to the monitors. The monitoring of the system is performed adopting several GNU/Linux tools such as *ganglia*, the power meter *likwid*, and a per-core temperature driver with an installation of the GNU/Linux sensor library *lm-sensors*.

There is a trade-off concerning power consumption and heat. A middle QoS point exists between the non-QoS performance and power save points. Regarding relative values to QoS execution, the power consumption is reduced by 21% for one application and 24 % for two applications concerning the performance execution.

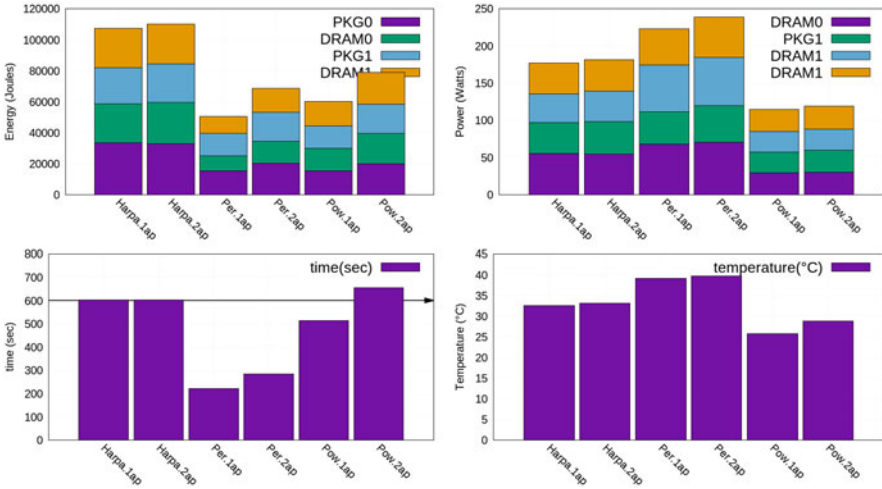


Fig. 12.9 Top-left: Total energy in joules for the six situations (3 governors with the two models (uncertainty model alone and uncertainty with WIA)), the energy can be divided in energy of the socket packages more the energy of the two modules DRAM. Top-right: Presents similar situations but power results in Watts. Bottom-left: Presents the execution time for the six situations, Bottom-right: The mean temperature of the 16th cores just before the end of the execution

Concerning temperature are 7 °C lower for both configurations than performance and 7 and 4 °C higher for one and two applications than power save execution. Because the execution time in performance execution takes 222 and 283 s, and there is no QoS monitoring, an overhead in resource utilisation is created. Therefore, the energy consumption of the system is lower than with QoS (HARPA-OS) execution.

12.4.2 Time Predictability with HARPA-OS

The main aim of the presented experiments is to show the reliability improvement in terms of time predictability [13]. The first experiment presented is only one example of possible execution of the model alone and with constraints (i.e. WIA model). But what can happen in terms of a reliable QoS for several running iterations, monitoring the power, energy and temperature of one system?

12.4.2.1 Power and Energy Monitors

Figure 12.10 presents the results of running the model one thousand iterations for 1 min and 1.2K MC samples. The figure presents results of running only the

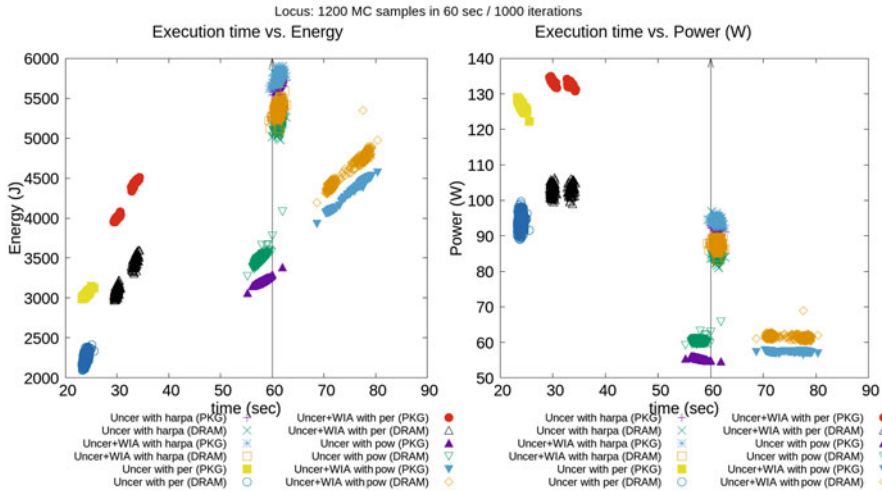


Fig. 12.10 Power and energy consumption vs. execution time in package (PKG) and memories (DRAM). Execution of 1K iterations of uncertainty model and uncertainty+WIA models. The iterations runs in three modes, two non-QoS named performance *per* and powersave *pow*, and one with QoS named *HARPA*

uncertainty model (*uncer*) and also running both models (*uncer+WIA*). The WIA starts execution of a random δ_0 from 0 to 15 s and performs two iterations (δ time).

Figure 12.10 shows the execution time vs. energy and power consumption in the chip package and DRAMs memories. The figure also reveals that all the iterations provide the results before or on the deadline of 1 min. Only the uncertainty with the WIA as a constraint in power save (*uncer+WIA with pow*) finalises after the term. The experiment reinforces the idea that there is a middle point regarding power consumption between the two GNU/Linux governors. The figure shows that with HARPA-OS activated the system has an overhead of 9–28% (J) with respect to the performance governor.

12.4.3 Execution of the Application Scenarios on the Platform

In this subsection, we run the uncertainty module on 16 nodes of an HPC cluster. The cluster presented has 18 blades. From all possible resources, we use 16 for our experiments, leaving two blades as a backup, log file system. The uncertainty module uses a hybrid OpenMP and MPI approach [22] to distribute the computations to multiple nodes.

Given that the performance requirements of the application are time-variable (e.g. low when sunny, intermediate/critical when rainy), the HPC centre may only allocate some computing nodes to the application and use the remaining ones

Table 12.1 For each experiment, number of Monte Carlo samples to be performed by the instance that models each catchment

Experiment	Thousand of MC samples to perform			
	C ₁	C ₂	C ₃	C ₄
α_1	1.5	1.5	1.5	1.5
α_2	1.5	3.5	5.0	7.0
α_3	7.0	7.0	7.0	7.0
β_1	3.5	3.5	3.5	3.5
β_2	7.0	7.0	7.0	7.0
β_3	3.5	7.0	12.0	15.0
γ	80.0 between all catchments			

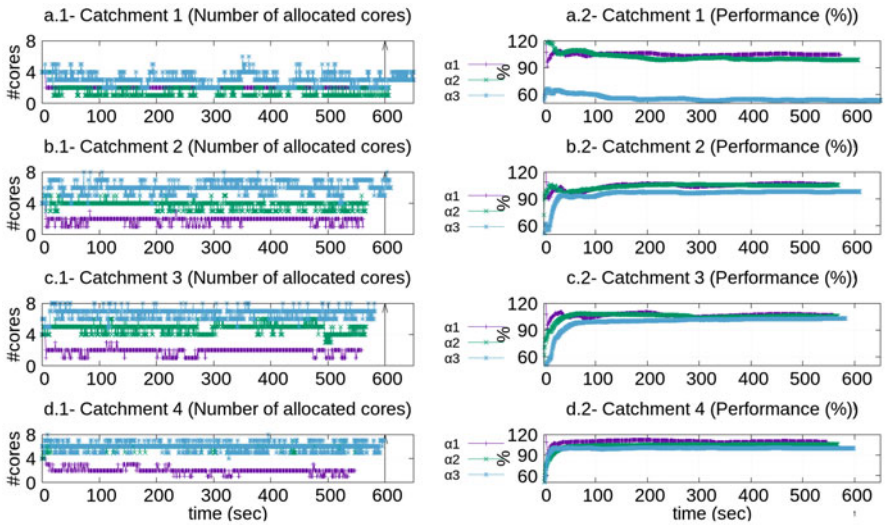


Fig. 12.11 Standard operation: light or no precipitation, low water level, (α experiments)

to execute other applications. Therefore, we performed our experiments in three different configurations: in the first one, we used only one node (i.e. standard operation). In the second, we used two nodes (i.e. intermediate operation). In the third, we used 16 blades of the cluster (i.e. critical operation).

Table 12.1 presents the set of experiments performed in the cluster. The experiments tagged α refer to the single-node scenario, while those tagged with β and γ respectively refer to the dual-node and 16 nodes are used from the entire cluster scenarios.

12.4.3.1 One-Node Configurations

Figures 12.11a.1, 12.11b.1, 12.11c.1 and 12.11d.1 show the number of cores allocated during the 10 min execution for each catchment, while Figs. 12.11a.2, 12.11b.2, 12.11c.2 and 12.11d.2 show the monitored performance

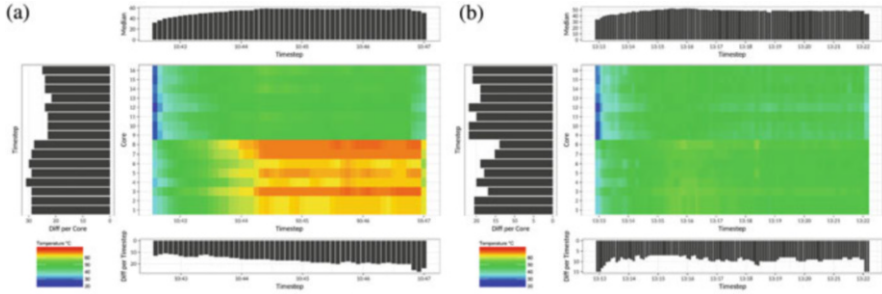


Fig. 12.12 Thermal map per core for a 10 min execution in one node. (a) Shows best-effort performance GNU/Linux governor (12K in total). (b) Shows the heat map in the case of HARPA PerDeTemp runtime with GNU/Linux performance governor (12K in total)

gap (g_p equations (12.2)) (as presented in Eq. (12.3), similar to (12.2), but in this case, the closer to 100% the gap is, the better the performance).

$$gap_{\text{performance}} = \frac{\text{throughput}_{\text{current}}}{\text{throughput}_{\text{ideal}}} (\%) \quad (12.3)$$

As shown by the experimental results, the number of allocated resources gets higher as the number of samples of MC to be performed increases. The allocation of resources is satisfactory for α_1 and α_2 scenarios, but not for α_3 . In Fig. 12.11a.2, we can observe that the performance gap is below 100%, meaning that the resources required for this experiment are not enough. In this situation, a second node should be allocated.

Figure 12.12 presents the heat map of a single node when running an uncertainty module instance (12K MC samples) using a best-effort and just-in-time configuration, respectively. We obtained the heat maps by using the ganglia monitoring tool. In both figures, the X-axis represents time and the Y-axis represents the cores IDs.

- The *Diff per Core* (see the left part of the figures) is the difference in temperature per core c_i ($0 \leq i < 16$) during a period of time. In our case, the maximum difference of temperature of a core for a period of 10 min. Given a core c_i , and its maximum $t_{\text{max}}^{c_i}$ and minimum $t_{\text{min}}^{c_i}$ temperature, the *Diff per Core* i is $T_D^{c_i} = t_{\text{max}}^{c_i} - t_{\text{min}}^{c_i}$.
- The *Diff per Timestep* (the lower part of the figures) is the difference in temperature among all cores given a timestep. Given a time t and temperature T , the *Diff per Timestep* is $T_D^t = t_{\text{max}}^t - t_{\text{min}}^t$.
- The *Median* (the upper part of the figures) provides the median temperature per timestep. Given a time t , n equals the number of cores, and temperature T , the *Median* is $T_{\text{med}} = \sum_{c=0}^{c=n-1} \frac{T_c^t}{n}$.

As can be seen in Fig. 12.12a (best-effort execution with CPUfreq performance governor) there is a hotspot in socket 0 [cores 0..7], which, as already shown in

Fig. 12.6, is the socket that is further from the fan. Conversely, Fig. 12.12b shows the execution with HARPA-OS-PerDeTemp. In this case, there are no hotspots and, thanks to the temperature-aware task migration performed by PerDeTemp, the temperature is perfectly distributed throughout all the available processing elements.

12.4.3.2 Two Nodes

Similarly, Fig. 12.13 shows the results for the dual-node configurations. In this case, β_2 has an equal number of samples to operate as α_3 , and, since in this case, we have two computing nodes at our disposal, the computation is performed without issues. But the limit of 10 min is exceeded according to Fig. 12.13. Hence, in this case, the resources are again not enough for all the scenarios: Fig. 12.13b.2 shows that the performance gap of β_3 is below 100%.

12.4.3.3 Multi-Node Results

In this section, we show and discuss the results of the execution of the multi-nodes scenarios. On the application side, we used an MPI implementation of the Uncertainty model to enable the possibility of launching the application on multiple nodes. Moreover, for each node an instance of HARPA-OS was running, to manage the resource assignment locally. Again, we compared the HARPA-OS distributed execution of the workload against the unmanaged configuration with

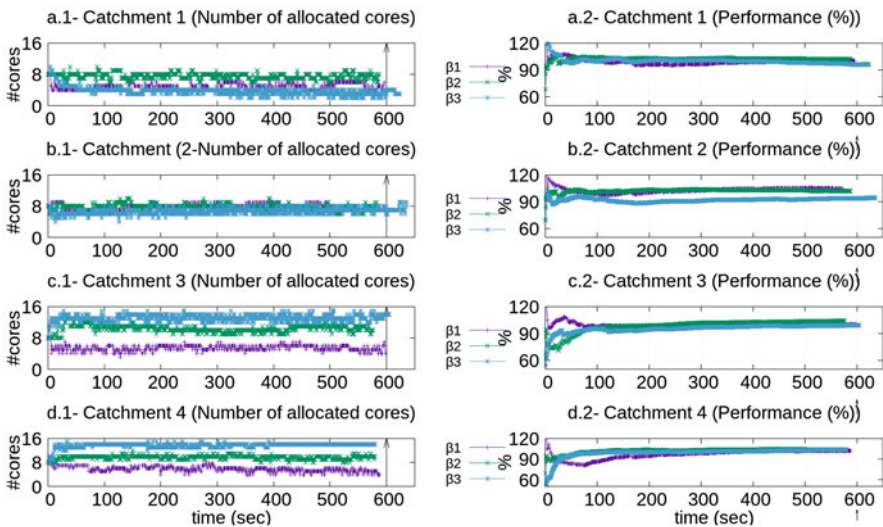


Fig. 12.13 Intermediate operation: intermediate execution, medium precipitation or water level warning threshold exceeded (β experiments) in two node execution

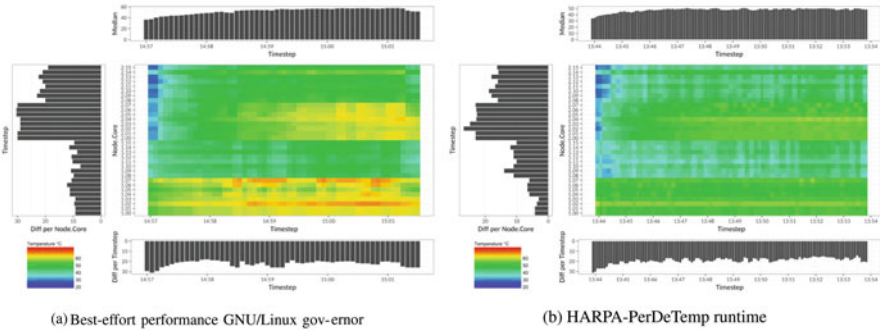


Fig. 12.14 Thermal map per core for 10 min execution of the two node scenario. Case (a) shows unmanaged execution (performance CPUfreq governor). Case (b) Shows HARPA-OS (PerDeTemp) runtime management (performance CPUfreq governor)

the Linux Completely Fair Scheduler (CFS) scheduler in charge of managing the CPU resource allocation, and the CPUfreq governor statically set (performance or power save). As we have done for the previous scenarios, the comparison includes considerations regarding performance, temperature of the CPU cores, power end energy consumption. In the last experiment of this set, we executed the γ scenario on 18 nodes (execution is performed in nodes 1 to 16 with MPI $n = 16$, nodes 17 and 18 remain idle) with a requirement of 80K MC samples. This time, all the instances terminated just-in-time, i.e., exactly at the deadline (Figs. 12.14, 12.15, and 12.16). Using the *likwid* power monitor, we monitored the power consumption in both the *jit* and the *be* configurations. Whereas just-in-time execution leads to a maximum consumption of 100 W, the best-effort execution leads to a peak of 160 W per node (presented in Fig. 12.17). Therefore, we saved around 44% in maximum power consumption, while nevertheless complying with the deadline.

12.5 Evaluating HARPA Impact on Hardware Reliability

The Mean Time Between Failure (MTBF) is a well-known reliability metric, expressing the average time it takes for a failure to occur. It is calculated using the following formula: $MTBF = (\text{Total device hours})/(\text{Total number of failures})$. The Failure Rate (FR) is the reciprocal of the MTBF:

$$FR = \frac{1}{MTBF} \quad (12.4)$$

Acceleration Factors

The increase of the temperature of the test environment to which the devices are subjected can accelerate most failure mechanisms. It is, therefore, possible to

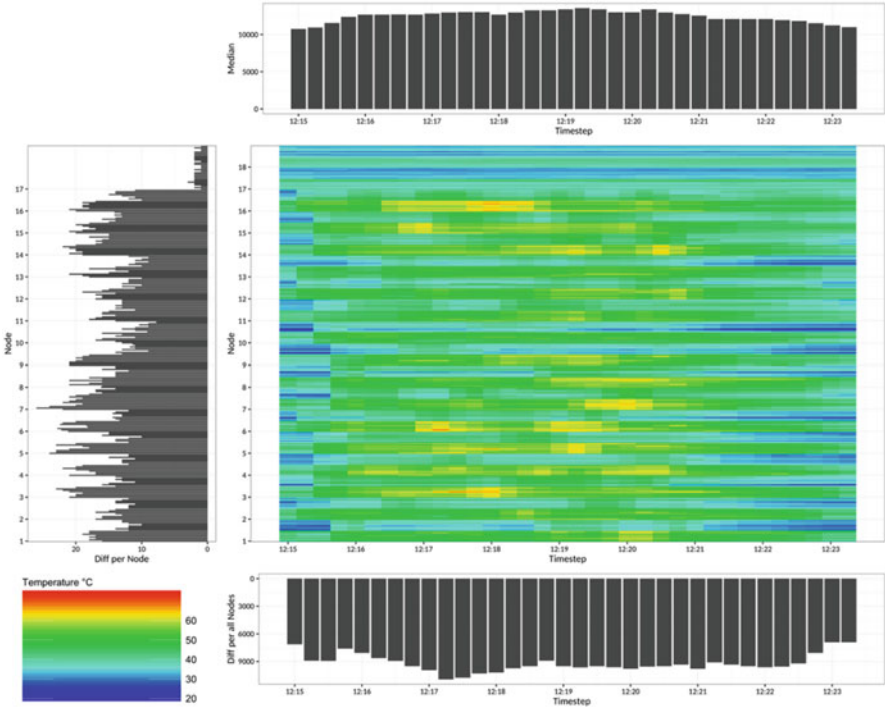


Fig. 12.15 HARPA-OS with PerDeTemp policy in 16 blades

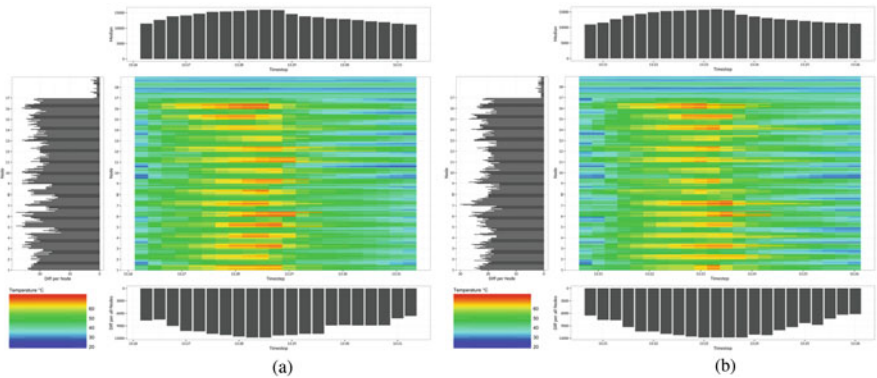


Fig. 12.16 Figures present hotspots (temperature peaks) even using the more conservative governor (power-save) in terms of power consumption. (a) Performance GNU/Linux governor in 16 blades. (b) Power-save GNU/Linux governor in 16 blades

perform relatively short tests which simulate many years of device stressing under more normal conditions. Obviously, it is necessary to have some measures of how greatly the failure mechanisms are accelerated. Suppose a device is stressed at a

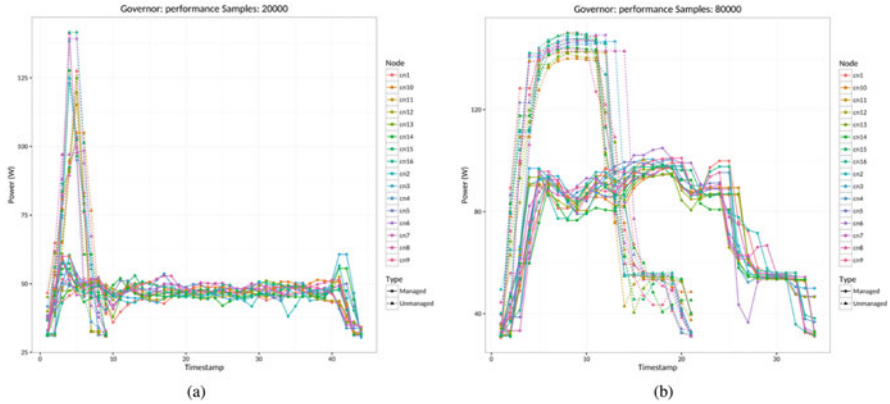


Fig. 12.17 Different behaviour in power consumption per blade. Performance governor presents peak consumption while PerDeTemp presents a decrease in peak performance. (a) Power consumption for 20K MC samples per blade. (b) Power consumption for 80K MC samples per blade

high temperature T_2 , for time t_2 . It is required to find the equivalent, t_1 , which would cause the device the same level of stress at a lower temperature T_1 . To this purpose, it is noted that if $T_2 > T_1$ then $t_2 > t_1$, we need to introduce an acceleration factor defined as follows:

$$A = \frac{t_1}{t_2} \tag{12.5}$$

hence, the ratio of the time at the lower temperature to that at the higher. Since the acceleration factor is a dimensionless ratio, it will be equal to the ratio of any other parameters proportional to the stress times. The most useful ratio is that of failure rates, which are inversely proportional to stress times if FR_1 is the failure rate at T_1 , and FR_2 the failure at T_2 then:

$$T_2 > T_1; FR_2 > FR_1; A = \frac{t_1}{t_2} \tag{12.6}$$

So if FR_2 is known, FR_1 is found simply as $FR_1 = A \times FR_2$. All that is now required is to find the value of the acceleration factor.

Temperature Acceleration

The High-Temperature-Bias (HTRB) test is an example of a temperature-accelerated test. It is usually found that the rates of the reactions causing device failure are accelerated with temperature according to the Arrhenius equation:

$$\text{Acceleration}_{\text{factor}} = e^{\frac{E_f}{k(\frac{1}{T_1} - \frac{1}{T_2})}} \tag{12.7}$$

where E_A = activation energy of the failure mechanism, k (Boltzmann constant = $8.6E-5$ eV/K), A , T_1 , T_2 are as defined above. The activation energy E_A is found experimentally, and is usually in the order of 1.0 eV, depending on the predominant failure mechanism. For the failure mechanisms for small silicon technologies, a value of 0.3 eV has been determined.

Mean Time Between Failures (MTBF)

The example is given for two sockets of 8 cores each in ANSELM. According to Sect. 12.3.2, both sockets have a difference of several °C when the chip is idle. We estimated the MTBF for 14K MC samples goal per 10 min. This situation can be the standard usage of the system. From the set of temperatures, we take the per core average minimum temperature T_1 and maximum temperature T_2 for the two execution models, the HARPA PerDeTemp policy and the best-effort performance GNU/Linux. The difference between socket 0 and socket 1 is illustrated in Sect. 12.3.3. The position of the fan on one side of the blade produces a gradient in temperature of around 10 (°C) degrees between sockets (Table 12.2).

$$\text{Relative}_{\text{acceleration factor}} = \frac{\text{Acceleration}_{\text{factor}(\text{best-effort})}}{\text{Acceleration}_{\text{factor}(\text{PerDeTemp})}} \quad (12.8)$$

According to Eq. (12.7), we have used a range of E_A values from 0.3 to 0.9 eV, the values which are taken as lower and upper limits, and the real value is in the range. Therefore, we computed the AF and the MTBF for the two sockets 0, 1 and the two policies. The results are exhibited in Table 12.3.

Table 12.2 Temperatures in Kelvin ($T_{(K)} = T_{(°C)} + 273.15$)

	Socket 0	Socket 1
<i>Best-effort</i>		
T_{\max}	340.525	324.525
T_{\min}	328.4	313.025
<i>PerDeTemp</i>		
T_{\max}	327.525	318.65
T_{\min}	321.775	310.775

Table 12.3 Acceleration factor (AF) and MTBF for two different cooling systems and two possible E_A

	0.3 eV	0.9 eV
AF <i>best-effort</i> , socket 0	1.46	3.11
AF <i>best-effort</i> , socket 1	1.48	3.27
AF <i>PerDeTemp</i> , socket 0	1.21	1.77
AF <i>PerDeTemp</i> , socket 1	1.32	2.30
MTBF <i>best-effort</i> , socket 0	0.69	0.32
MTBF <i>best-effort</i> , socket 1	0.67	0.31
MTBF <i>PerDeTemp</i> , socket 0	0.83	0.56
MTBF <i>PerDeTemp</i> , socket 1	0.76	0.44

Table 12.4 Relative MTBF best-effort vs. PerDeTemp

MTBF _{best-effort/PerDeTemp, socket 0}	0.83	0.57
MTBF _{best-effort/PerDeTemp, socket 1}	0.89	0.70

Reliability Trade-Offs

The presence of hotspots is known to have adverse effects on the Mean Time Between Failures of the system (MTBF) [19], and can be mitigated [23] through the migration of tasks from one processor to a different in a Chip MultiProcessor (CMP) system, if the processor heats up to a threshold value. The ageing acceleration factor depends on the difference (Δ) of temperature.

According to the MTBF (Table 12.4) estimation presented in [9], running the application in a HARPA-OS runtime with a PerDeTemp configuration improves the reliability of the system from 17% ($1 - \text{MTBF}_{\text{best-effort/PerDeTemp}} = 1 - 0.83, eV = 0.3$), Table 12.4) to 43% ($(1 - \text{MTBF}_{\text{best-effort/PerDeTemp}} = 1 - 0.57, eV = 0.9)$) in the case of poor cooling (socket 0). With a better cooled (socket 1), the MTBF for the best-effort relatively grows, but it is still 11% ($1 - \text{MTBF}_{\text{best-effort/PerDeTemp}} = 1 - 0.89, eV = 0.3$) to 30% ($1 - \text{MTBF}_{\text{best-effort/PerDeTemp}} = 1 - 0.7, eV = 0.9$) better if we manage the system with PerDeTemp (*jit*) instead of using the best-effort (*be*) policy.

12.5.1 Discussion of Applicability and Quality of the Results

The model used for the project is based on the Monte Carlo simulation. The Monte Carlo simulation is extensively used by the HPC community because it can be employed to generate massively parallel programs [6, 7]. The Monte Carlo simulation and its uncertainty are de facto fault tolerant since the algorithm can provide an answer even if some samples are lost. The rainfall–run-off model, its calibration [1, 27], and uncertainty must be executed every period that new data arrives from weather stations to provide as accurate results as possible. Hence, the QoS/time constraints allow relaxing of the execution requirements.

Since critical (alarm) situations are sporadic, instead of executing the model in best-effort, minimising the time but maximising the power consumption, running it with the QoS/time constraints is possible, as the results of the models must be provided before the deadline, otherwise it is too late. Our HPC models fall into the taxonomy of algorithms for urgent computing [28]. Therefore, the methodology can be reused for any program that falls under this classification. Thus, when the constraints are not very tight, it is possible to run the models allocating minimum resources and in the less degraded (healthier) parts of the system. HPC systems for urgent computing need very high availability and reliability. We have shown in this document the trade-offs between running the algorithm on the system in best-effort versus the Harpa-OS runtime. The balanced allocation of threads in the cores is taking into account which cores are more degraded, improving the reliability and hence expanding the lifetime of the system.

12.5.2 Conclusions

The chapter describes extensive testing and evaluation of representative applications of the HPC domain running with HARPA runtime support. The evaluation of the case studies has demonstrated the suitability of the techniques and methods developed in the HARPA project to both targeted compute domains (embedded and HPC). Benchmarking activities are performed as well, with reference platforms used to host these applications without HARPA framework. Validation of the HARPA environment verifies the fulfilment of our objectives concerning the HARPA environments functionality by monitoring and evaluating the performance, scalability and availability of adapted applications with a focus on efficiency in providing specified QoS levels in the presence of severe process variations. QoS levels are evaluated before and after the integration to enable the exact comparison of provided benefits. During the HARPA project, we have tried to assess the validation of the HARPA environment on HPC systems.

We have illustrated the advantages of running the HPC applications regarding availability and reliability. We have presented experiments that demonstrate the efficiency of providing specified QoS level even in the presence of degradation in the system (i.e. bubble). Yet still more work can be done concerning energy benefits. The current HPC systems are memory dominated. The power measurements are performed at the level of the socket, and inside the socket with the CPU, there are different layers of memory (L3, L2, L1 and LLC). According to our measurements, decreasing the Integrated Circuit (IC) power-supply pin V_{dd} with Dynamic Voltage and Frequency Scaling (DVFS) provides benefits regarding lowering the power, but not in reducing the global energy consumed during execution. This could be because the leakage power (static power) is dominant or has a high percentage of the consumption of the global system. Current monitors cannot provide us information about power consumption in the cores, only at the level of the socket. So, we think more experimentation is needed to ascertain what percentage of power consumption is due to the memory layer (mainly main memory) versus the CPUs in the socket. These tests could help us to disable or decrease the power consumption in memory parts that are underused, and in this way observe real energy savings.

In this chapter, we presented a framework that, based on the runtime-variable computational resources demand of applications, strives to minimize the resource usage of applications while making them comply with their deadlines. We call this practice *just-in-time execution* since the termination of the application is always as near as possible to the deadline. The framework can use the pool of unused [11] resources in multiple fashions, e.g., to swap allocated faulty processing elements with healthier ones, or to periodically assign to applications cool processing elements, hence evenly levelling the generated heat throughout the chip even in cases of asymmetrical cooling.

Acknowledgements This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science—LQ1602” and by the IT4Innovations infrastructure which is supported from the Large

Infrastructures for Research, Experimental Development and Innovations project “IT4Innovations National Supercomputing Center LM2015070”. The work was also supported by the European Union FP-7 program through the HARPA project (grant no. 612069).

References

1. Abulohom, M. S., Shah, M. S., & Ghumman, A. R. (2001). Development of a rainfall-runoff model, its calibration and validation. *Water Resources Management*, 15(3), 149163.
2. Bahremand, A., & De Smedt, F. (2008). Distributed hydrological modeling and sensitivity analysis in Torysa watershed, Slovakia. *Water Resources Management*, 22(3), 293408.
3. Bellas, P., Massari, G., & Fornaciari, W. (2012). A RTMR proposal for multi/many-core platforms and reconfigurable applications. In *ReCoSoC*.
4. Bellasi P., Massari, G., & Fornaciari, W. (2015). Effective runtime resource management using linux control groups with the barbequertm framework. *ACM Transactions on Embedded Computing Systems*, 14(2), 39:139:17.
5. Beven, K. (2012). *Rainfall-runoff modelling*. New York: John Wiley & Sons.
6. Billinton, R., Allan, R.N., IEEE Power Engineering Society. Power Engineering Education Committee, & IEEE Power Engineering Society. Power system Engineering Committee (1989). Reliability assessment of composite generation and transmission systems. IEEE Power Engineering Society Tutorial, 90EH0311-1-PWR.
7. Billinton, R., & Li, W. (1994). *Reliability assessment of electric power systems using Monte Carlo methods*. New York: Plenum Press.
8. Bolchini, C., Carminati, M., Gribaudo, M., & Miele, A. (2014). A lightweight and open-source framework for the lifetime estimation of multicore systems. In *IEEE 32nd International Conference on Computer Design (ICCD)*.
9. Ellerman, P. (2012). Calculating reliability using FIT & MTTF: Arrhenius HTOL model. In MICROSEMI, Technical Report.
10. Ganeshpure, K., & Kundu, S. (2014). Performance-driven dynamic thermal management of mpso based on task rescheduling. *ACM Transactions on Design Automation of Electronic Systems*, 19(2), 11:1–11:33.
11. Hardy, D., Sideris, I., Ladas, N., & Sazeides, Y. (2012). The performance vulnerability of architectural and non-architectural arrays to permanent faults. In *MICRO* (pp. 48–59).
12. Harpa harnessing performance variability fp7 project (2013). <http://www.harpa-project.eu>
13. Hartman, A. S., & Thomas, D. E. (2012). Lifetime improvement through runtime wear-based task mapping. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'12* (pp. 13–22). New York: ACM.
14. Golasowski, M., Litschmannova, M., Kuchar, M., Podhoranyi, M., & Martinovic, J. (2015). Uncertainty modelling in rainfall-runoff simulations based on parallel Monte Carlo method. *International Journal on Non-standard Computing and Artificial Intelligence NNW*, 25(3), 267–286.
15. Martinovic, J., Kuchar, S., Vondrak, I., Vondrak, V., Sir, B., & Unucka, J. (2010). Multiple scenarios computing in the flood prediction system FLOREON. In *European Conference on Modelling and Simulation* (pp. 182–188).
16. Massari, G., Libutti, S., Portero, A., Vavrik, R., Vondrak, V., & Fornaciari, W. (2015). Harnessing performance variability: A HPC-oriented application scenario. In *Euromicro Conference on Digital System Design (DSD)*, Funchal, Madeira, Portugal.
17. Meesuka, V., Vojinovica, Z., Mynetta, A. E., & Abdulla, A. F. (2015). Urban flood modelling combining top-view LiDAR data with ground-view SfM observations. *Advances in Water Resources*, 75, 105–117.

18. Nash, J., & Sutcliffe, J. (1970). River flow forecasting through conceptual models part I. A discussion of principles. *Journal of Hydrology*, 10(3), 282-290. [https://www.doi.org/10.1016/0022-1694\(70\)90255-6](https://www.doi.org/10.1016/0022-1694(70)90255-6). *Procedia Computer Science*
19. National Semiconductor. (2000). Understanding integrated circuit package power capabilities. Available: www.national.com
20. Oh, D., Kim, N. S., Chen, C. C. P., Davoodi, A., & Hu, Y. H. (2010). Runtime temperature-based power estimation for optimizing throughput of thermal-constrained multi-core processors. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference, ASPDAC'10*, Piscataway (pp. 593–599). New York: IEEE Press.
21. Portero, A., Kuchar, S., Vavrik, R., Golasowski, M., & Vondrak, V. (2014). System and application scenarios for disaster management processes, the rainfall-runoff model case study. In *CISIM* (pp. 315–326).
22. Portero, A., Sevcik, J., Golasowski, M., Vavrik, R., Libutti, S., Massari, G., et al. (2016). Using an adaptive and time predictable runtime system for power-aware HPC-oriented applications. In *IGSC 2016* (pp. 1–6).
23. Powell, M. D., Goma, M., & Vijaykumar, T. N. (2004). Heat-and-run: leveraging SMT and CMP to manage power density through the operating system. In *ASPLOS*.
24. Qiua, L., Dua, Z., Zhua, Q., & Fane, Y. (2017). An integrated flood management system based on linking environmental models and disaster-related data. *Environmental Modelling & Software*, 91, 111–126.
25. Sacerdoti, F. D., Katz, M. J., & Massie, M. L. & Culler, D. E. (2003). Wide area cluster monitoring with Ganglia. In *CLUSTER* (Vol. 3, pp. 289–289).
26. Treibig, J., Hager, G., & Wellein, G. (2010). Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *Proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW)* (pp. 207–216). New York: IEEE.
27. Vavrik, R., Theuer, M., Golasowski, M., Kuchar, S., Podhoranyi, M., & Vondrak, V. (2015). Automatic calibration of rainfall-runoff models and its parallelization strategies. *AIP Conference Proceedings*, 1648, 830014. <https://aip.scitation.org/doi/abs/10.1063/1.4913040>
28. Yoshimoto, K. K., Choi, D. J., Moore, R. L., Majumdar, A., & Hocks, E. (2012). Implementations of urgent computing on production HPC systems. *Procedia Computer Science*, 9, 1687–1693.